ProFlex: A Probabilistic and Flexible Data Storage Protocol for Heterogeneous Wireless Sensor Networks

Guilherme Maia, Daniel L. Guidoni, Aline Carneiro Viana, André L. L. Aquino, Raquel A. F. Mini, Antonio A. F. Loureiro

▶ **To cite this version:**

Guilherme Maia, Daniel L. Guidoni, Aline Carneiro Viana, André L. L. Aquino, Raquel A. F. Mini, et al.. ProFlex: A Probabilistic and Flexible Data Storage Protocol for Heterogeneous Wireless Sensor Networks. [Research Report] RR-7695, INRIA. 2011. inria-00610620

HAL Id: inria-00610620
https://inria.hal.science/inria-00610620

Submitted on 22 Jul 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

ProFlex: A Probabilistic and Flexible Data Storage Protocol for Heterogeneous Wireless Sensor Networks

Guilherme Maia — Daniel L. Guidoni — Aline C. Viana — André L. L. Aquino — Raquel A. F. Mini — Antonio A. F. Loureiro

N° 7695

July 2011

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--7695--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SACLAY - ÎLE-DE-FRANCE

ProFlex: A Probabilistic and Flexible Data Storage Protocol for Heterogeneous Wireless Sensor Networks

Guilherme Maia , Daniel L. Guidoni , Aline C. Viana , André L. L. Aquino , Raquel A. F. Mini , Antonio A. F. Loureiro

Thème COM — Systèmes communicants
Équipes-Projets Hipercom

Rapport de recherche n° 7695 — July 2011 — 20 pages

Abstract: This paper presents ProFlex, a proactive data distribution protocol for heterogeneous wireless sensor networks (HWSNs). ProFlex guarantees robustness in data retrieval by intelligently managing data replication among selected storage nodes in the network. Contrarily to related protocols in the literature, ProFlex considers the resource constraints of sensor nodes and constructs multiple data replication structures, which are managed by more powerful nodes. Additionally, ProFlex takes profit of the higher communication range of such powerful nodes in the network and use the long link to improve data distribution. When compared with Supple – a related protocol, we show by simulation that Proflex increases the network resilience under failures circumstances, decreases the overhead of transmitted messages, and decreases the number of hops to find a specific data in the network.

Key-words: Distributed Data Storage, Heterogeneous Sensor Networks

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université
4, rue Jacques Monod, 91893 ORSAY Cedex
Téléphone : +33 1 72 92 59 00

ProFlex: un protocole de stockage flexible et probabilistes de données pour les réseaux sans fil de capteurs hétérogènes

Résumé :

Cet article porte sur la proposition d'un protocole de données proactif de distribution pour les réseaux de capteurs sans fil hétérogènes (HWSNs). Notre protocole, ProFlex, garantit la robustesse de la récupération des données grâce à sa gestion intelligente de la réplication des données entre les nuds de stockage sélectionnés dans le réseau. Contrairement à d'autres protocoles dans la littérature, ProFlex considère les contraintes de ressources de capteurs et construit plusieurs structures de réplication des données, qui sont gérés par des nuds plus puissants. En outre, ProFlex profite de la meilleure communication radio de ces nuds plus puissants et utilise ces longues portée pour améliorer la distribution des données. Nous avons comparé ProFlex avec le protocole Supple et nous avons montré par simulation que Proflex augmente la résilience du réseau, même dans des circonstances des pertes de messages, diminue la surcharge de messages transmis, et diminue le nombre de sauts nécessaire pour trouver un ensemble de données spécifiques dans le réseau.

Mots-clés : Stockage distribué de données, réseaux de capteurs hétérogènes

1 Introduction

The deployment of new Wireless Sensor Network (WSN) applications (e.g., health monitoring of patients, environment sensing, and factory monitoring systems), which operate unattended for a long period of time and generate a considerable amount of data, poses several challenges. In such applications, one of the major challenge is *where and how to store and retrieve the sensed data*.

For this, different distributed data storage protocols have been proposed in the literature as a way to increase resilience to failures and, hence, decrease possible losses of sensed data [13, 14, 17, 9, 16, 4]. A distributed approach has the advantage of not having a single point of failure, since the sensed data is distributed among the network nodes. Nevertheless, several problems arise in this context, such as *how to efficiently store the sensed data to be later retrieved* and *how much data a sensor node should store*. The first problem is related to resilience and corresponds to the distribution of the sensed data over the network, whereas the second one is related to resource management and corresponds to how much data from other nodes a node should store.

To overcome the aforementioned problems, some studies proposed the use of more powerful nodes to perform data storage [13, 1]. In this scenario, only these powerful nodes are responsible for storing all sensed data, since the assumption is that those nodes have no memory limitation. Nevertheless, the use of more powerful nodes does not overcome the problem of data losses, since these nodes still can fail. To increase the network resilience to failures, a possible approach is to replicate a given data and keep it at different nodes. Moreover, in the presence of a mobile sink, a good setup on the number of replicas at nodes might enable the sink to get a representative view of the entire network data by only visiting a small percentage of nodes [16, 4]. Hence, the use of more powerful nodes as data storage nodes also requires replication mechanisms to deal with resilience.

In this work, we propose the use of powerful nodes to perform distributed data storage in Heterogeneous Sensor Networks (HSNs). However, instead of using the extra memory features of these nodes, we take profit of their powerful communication range and use the long link to improve data distribution. The proposed protocol, called ProFlex, was designed to be aware of the heterogeneous topology. ProFlex is divided into three phases (Section 3). The first phase is a *tree construction* initiated by all powerful nodes. These trees will be used to route the collected packets from sensor nodes to their selected storage nodes. The second phase performs the *importance factor distribution*, which will be used in the last phase to determine the probability of a node to become a storing node. The last phase is the *data distribution process*, which uses the tree topology and the importance factor to decide whether a sensor node should store the data. Due to their similarities, we evaluate and compare ProFlex with the Supple [4] protocol (Section 4). Simulation results show that using a heterogeneous network topology, ProFlex increases the network resilience under failures circumstances, decreases the overhead of transmitted messages, and decreases the number of hops to find a specific data in the network. In addition, we discuss the related work and present the system model in Section 2, and conclude the paper and discuss the future work in Section 5.

2 Background

2.1 Data Storage Protocols

In the literature, there are some proposals for data storage in WSNs. We discuss some of them in the following.

Sheng et al. [13, 14] study the data storage placement in WSNs to deal with the traditional problem of traffic overhead and, consequently, high energy consumption of nodes closer to the sink. To overcome this problem, they propose two network models. The first one considers a tree topology rooted at the sink and a subset of afterward selected storage nodes, storing data collected by their descendants in the tree. In the second model, a tree topology is constructed after the deployment of the storage nodes, whose positioning is obtained from a linear programming optimization. Nevertheless, node failures are not considered in both models, which might result in the loss of all data collected by storage nodes' descendants.

Anastasi et al. [1] and Tseng et al. [15] investigate the use of data mules (mobile nodes with higher hardware capabilities) to collect and store data in wireless sensor networks. The drawback of this approach is the problem of data mule failures, in which case all collected data might be lost. Also, this approach has high latency to collect the sensed data, since the mule node must be close enough to the sensor node to collect the data.

To overcome the problem of storage node failures, proactive data replication strategies have been proposed in the literature. The goal of such approaches is to replicate sensed data to a selected subset of nodes [17, 9, 16, 4], which can be later retrieved by mobile sinks. Such approaches consider predictable [17, 9] or uncontrolled [16, 4] mobility of the sink nodes.

More related to our approach is the Supple protocol [4] — a flexible probabilistic data dissemination protocol for WSNs that considers static or mobile sink nodes. The Supple protocol has three phases: tree construction, weight distribution, and data replication. The first phase relies on a binary tree construction by a central sensor node of the sensing area (e.g., the sink node). The central sensor node is responsible for receiving and replicating the collected data in the network. The second phase assigns weights to nodes, which represent the probability of a node storing a data. Supple uses the hop distance of a node to the central node to calculate this probability. In the last phase, the sensor nodes send their data to the central node and this node replicates each data $r(v)$ times using the tree infrastructure and according to its storage probability. The number of replicas depends on the weights and the amount of data each node is allowed to store. The authors claim that a mobile sink node visiting a small fraction of nodes, i.e., about $2.3\sqrt{n}$, for a total of n nodes, can retrieve all the generated data in the network. Moreover, thanks to $r(v)$, the failure of a small number of nodes will not cause data losses. However, the Supple protocol does not consider the problem of finding a good positioned central node as well as the problem of energy consumption and traffic overload through the nodes closer to the central node.

Our protocol ProFlex deals with proactive data distribution in WSNs and with the Supple's drawbacks. ProFlex guarantees robustness in data retrieval by intelligently managing data replication among selected storage nodes and considers sensor nodes' resource constraints by using multiple replication struc-

tures. Although the described drawbacks, we evaluate ProFlex's performance compared Supple since it is the most similar protocol to ours.

2.2 System Model

The main adopted assumptions are detailed hereafter.

Nodes. We consider that there is a large number (n) of sensor nodes scattered on a given geographic area for collecting data or monitoring events. All sensors are uniquely identified and can be of two types. The first one, named *L-sensor* for low-end sensor, is a node with limited resources, including processor, storage, communication and power resources. The second type, named *H-sensor* for high-end sensor, is a node with more sophisticated resources. Thus, *H-sensor* nodes have improved processing, storage, battery and communication power when compared to *L-sensor* nodes. A question that might arise is: *Why not design a sensor network comprised of just H-sensor nodes?* Whereas *H-sensor* nodes are more powerful when compared with *L-sensor* nodes, the latter are much less expensive. Hence, it is assumed the network is composed of n_l *L-sensor* nodes, and n_h *H-sensor* nodes, where $n = n_l + n_h$ and $n_l \gg n_h$. It is worth noting that in this work, we assume that only *L-sensor* nodes are data producers, so *H-sensor* nodes do not perform data sensing. Moreover, nodes later selected as a storage node are provided with a *partial view* v regarding some other nodes, including itself. We define S as the *storing nodes set*. Therefore, each node may act as a storage node for some other nodes, but not for all of them. Due to the limited buffer of *L-sensor* nodes, power-aware compression [12] and reduction algorithms [2] may be employed.

Communication. We consider a connected network topology along the time. Given the expected network lifetime, we can estimate the amount of sensor nodes to achieve this goal. A *L-sensor* node i can communicate with another node j (*L-sensor* or *H-sensor*) that is inside its communication radius r_1, i.e., the distance between i and j should be less than or equal to r_1 ($d(i,j) \leq r_1$). *H-sensor* nodes are equipped with two radios, each one with a different frequency and a different communication radius (r_1 and r_2, $r_2 \gg r_1$). It is also assumed that radio frequencies do not interfere with each other. A *H-sensor* node can communicate with both *L-sensor* and *H-sensor* nodes inside communication radius r_1 and r_2, respectively. Finally, the network is modeled as a graph $G = (V, E)$ where $V = V_L + V_H$ is the set of *L-sensor* and *H-sensor* nodes, and $E = E_L + E_H$ models their neighboring links. We assume the long range links r_2, among *H-sensor* nodes, are selected in such a way to yield a network with a small average path length, as presented in [8]. Therefore, a message in a network with this characteristic needs less hops to reach its destination.

Initial knowledge. Initially, a node $i \in V$ only knows its identity, which is unique, its sensor type (*L-sensor* or *H-sensor*), and a parameter $I(i)$ that defines its *importance factor* in the network ($I\colon S \rightarrow \{0,1\}$, $S \subseteq V$, called the importance factor function). Importance factors are initially assigned to nodes based on an external criterion. It determines the nodes in the network responsible for storing data. If the criterion is the sensor location, only nodes at a specified location will be used as storing nodes and will have $I(i) = 1$. It may also be desired to choose as storing nodes only *H-sensor* nodes, since they

are more powerful than *L-sensor* nodes. In this scenario, *H-sensor* nodes will have $I(i) = 1$ and *L-sensor* nodes will have $I(i) = 0$. On the other hand, if all nodes can be uniformly selected as storing nodes, then all nodes in the network will have $I(i) = 1$. In this work, we choose as storing nodes only *L-sensors*, hence for all *L-sensor* nodes $I(i) = 1$ and for all *H-sensor* nodes, $I(i) = 0$. In summary, the distribution of the importance factors among nodes will define the storing nodes set S.

3 Proposed Protocol

In this section, we present ProFlex, a Probabilistic and Flexible Data Storage Protocol for Heterogeneous Wireless Sensor Networks. ProFlex provides the capability for each node in a network to independently determine its own group of storage. Although requiring a predefined selection criterion, this storing selection procedure enables its combination with data gathering strategies based both on static and mobile sinks. Initially, we introduce the major steps of the protocol and how it allows a flexible data distribution to a previously specified subset of storing nodes in a WSN. Then, we present a general discussion about the main characteristics and possible drawbacks of the algorithm when compared with Supple [4].

3.1 ProFlex Algorithm

The ProFlex algorithm is comprised of three phases. The first one builds a tree-based routing structure starting with the *H-sensor* nodes. The second phase distributes the *importance factor*, in which the *L-sensor* nodes forward their importance factor value towards the *H-sensor* nodes operating as root of their tree. Finally, in the third data *L-sensor* nodes forward data to their *H-sensor* root nodes to be later replicated among nodes in the *storing nodes set* S. Algorithm 1 presents a general overview of ProFlex.

Algorithm 1: General principle of ProFlex

Input: Graph $G = (V, E)$
Input: Storing nodes set $S \subseteq V$
Input: Importance factor function $I : S \longrightarrow \mathbb{N}$
Input: Storing function $X : V \longrightarrow \{0, 1\}$

1. Construction of the trees $T_H(G)$ from G
2. Propagation of the importance factor and number of storing nodes in each tree
3. **foreach** *node* $i \in V_L$ **do**
   Send $data(i)$ to the root of $T_H(G)$
4. The root propagates each $data(i), r(v)$ times according to the probabilities induced by the importance factors over the storing nodes set

3.1.1 Tree Construction

The first step of ProFlex is a tree construction initiated by all *H-sensor* nodes in the network. Let $G = (V, E)$ be the graph representing the network, $T_H(G)$ the trees that aggregate the shortest paths from each *L-sensor* node to the closest *H-sensor* node. In this work, a shortest path means the number of hops, but any other metric can be used, i.e., delay, capacity, etc. It is worth mentioning that although each *H-sensor* node builds a tree rooted at itself, each *L-sensor* node will belong **only** to the tree rooted at the closest *H-sensor* node. Hence, during the tree construction, when a *L-sensor* node receives several *H-sensor*'s messages, it will update its local information and forward the message further, only if the message is from a closer *H-sensor* node. Otherwise, the node will simply discard the message.

For comparison reasons with the Supple protocol (Section 4), we consider the use of a binary tree as a routing structure. We also use in our performance analysis the PeerNet [5] protocol to build the binary tree. ProFlex supports, however, any other routing structure.

3.1.2 Importance Factor Distribution

In ProFlex, all nodes of the *storing nodes set* have an *importance factor* assigned by the function $I\colon S \to \{0, 1\}$. The importance factor assigned to a particular node i dictates if node i will play the rule of storing data for other nodes ($I(i) = 1$) or not ($I(i) = 0$) and will be later used at the computation of node i's storing probability.

The importance factor function of nodes is defined by a selection criterion. Although ProFlex allows the use of any selection criterion, for comparison reasons with the Supple protocol, we use the uniform selection criterion. Thus, $S = V_L$ and for all nodes $i \in S, I(i) = 1$; if $i \notin S$, $I(i) = 0$. This gives to all nodes in S equal chance of being selected as storing nodes. Note that the storing nodes set S is **only** composed by *L-sensor* nodes, therefore no *H-sensor* node will store any packet. It is straightforward to notice that due to their better resource capabilities, *H-sensor* nodes could also be used as storage nodes and have a greater importance factor than *L-sensor* nodes. Thus, the former would store more data than the latter, but we choose not to do so. Such a decision is backed by the fact that in this work, we only intend to leverage the characteristics inherent to heterogeneous WSNs and not the power of *H-sensor* nodes themselves.

In existing protocols [4, 3, 16], the *partial view size* $|v|$ (i.e., maximum number of allowed stored packets at a given node) is a statically configured parameter. It is configured considering a uniform distribution of the data and based on the size of the storing nodes set in the considered replication structure. By doing this, they ensure that there will be enough space to store the data generated by the network. In particular, in Supple, a unique tree-based replication structure is considered and, if uniform selection criterion is used, the set size of the storing nodes will be equivalent to the number of nodes in the tree, i.e., n nodes. Instead, ProFlex uses several replication structures at the data distribution phase, which are given by the multiple constructed trees. Thus, *each tree defines different storing nodes and S sizes.* This requires a dynamic configuration of the partial view size of nodes per replication structure.

Algorithm 2: Importance factor and number of storing nodes distribution algorithm

Input: Trees $T_H(G)$
Input: Storing nodes set $S \subseteq V$
Input: Importance factor function $I\colon S \longrightarrow \mathbb{N}$
Input: Storing Function $X\colon V \longrightarrow \{0,1\}$

foreach *node* $i \in T_H(G)$ **do**
  create $(I_l(i), I(i), I_r(i), X_l(i), X(i), X_r(i))$
foreach *node* $i \in T_H(G)$ *in a breadth-first search starting from the leaves* **do**
  if $j =$ *left child of* i **then**
    $I_l(i) = I_l(j) + I(j) + I_r(j)$
    $X_l(i) = X_l(j) + X(j) + X_r(j)$
  if $k =$ *right child of* i **then**
    $I_l(i) = I_l(k) + I(k) + I_r(k)$
    $X_l(i) = X_l(k) + X(k) + X_r(k)$
foreach *node* $i \in V_H$ **do**
  $|S_i| = X_l(i) + X(i) + X_r(i)$
  Send $|S_i|$ to *H-sensor* neighbors
foreach *node* $i \in V_H$ **do**
  $|S^i_{aggr}| = |S_i| + \sum_{j \in N(i)} |S_j|$
foreach *node* $i \in V_H$ **do**
  $|v| = \sqrt{|S^i_{aggr}|}$

Viana et al. [4] show that for a partial view of size $|v|$ and a network with n data producers, the storing nodes set S must contain at least $\Theta(\frac{n}{v} \ln n)$ nodes in order to guarantee with high probability a good storage of all n collected data. On the other way, the partial view size must be $v \geq \frac{n}{|S|} \ln n$. Viana et al. also show that a partial view size $|v| = \sqrt{n}$ provides a good compromise between resilience and sensors' resource consumption when $|S| = n$. Notice that here, the partial view size depends on the set size $|S|$ of the storing nodes and the number of data producers n in the replication structure. Since ProFlex uses several replication structures rather, the partial view size $|v|$ of storing nodes will be different for each tree t. As discussed in Section3.1.3, a *H-sensor* node h stores in its tree t_h all data produced by nodes belonging to its tree and by nodes belonging to the *neighboring trees*. Neighboring trees are trees rooted at *H-sensor* neighbors of the *H-sensor* root node h, denoted by $N(h)$.

For the special case where $n = |V_L|$, a *H-sensor* node only needs to know the set size of the storing nodes of its tree and of the neighboring trees. Consider the storing function $X\colon V \to \{0,1\}$, where $X(i) = 1$ if $I(i) \neq 0$; or $X(i) = 0$ if $I(i) = 0$. Algorithm 2 defines the set size of the storing nodes and the importance factor. The main idea behind this algorithm is to initialize each node $i \in V$ with $(I_l(i), I(i), I_r(i), X_l(i), X(i), X_r(i))$, where $I_l(i)$ and $X_l(i)$ (similarly to third component $I_r(i)$ and sixth component $X_r(i)$) are the importance factor and number of storing nodes of the left (similarly to right) subtree of i, and $I(i)$ and $X(i)$ are the importance factor and the value of the storing function of the node i in the set of storing nodes, respectively. Note that $I_l(i)$ (similarly to

$I_r(i)$) is the sum of all importance factors of nodes in the left (right) subtree of node i.

When the *H-sensor* node h knows the set size of the storing nodes $|S_h| = X_l(h) + X_r(h)$ in its tree, it will forward this value to its *H-sensor* neighbors. Eventually, a *H-sensor* node h will also receive the set size of the storing nodes from its neighboring *H-sensor* nodes. Finally, a *H-sensor* node h calculates the set size of the aggregated storing nodes $|S^h_{aggr}|$, i.e., its own set size of the storing nodes plus the set sizes of the storing nodes at its neighboring trees: $|S^h_{aggr}| = |S_h| + \sum_{j \in N(h)} |S_j|$. Based on this information, a *H-sensor* node h calculates the partial view size $|v| = \sqrt{|S^h_{aggr}|}$ for storing nodes in its tree.

3.1.3 Data Distribution

Algorithm 3: Data distribution algorithm

Input: Tree $T(G)$
Input: $(I_l(i), I(i), I_r(i), X_l(i), X(i), X_r(i))$ for each $i \in V$

```
// Upward tree
foreach node i ∈ V do
    Send data to its parent
if node i receives data from child then
    if i ≠ H-Sensor node then
        Forward data to parent
    if i ∈ V_H then
        Forward data to H-sensor neighbors
        r(v) ← NumberOfReplicas(data)
        Call ForwardData(data) r(v) times
// Downward tree
if node i ∈ V_H receives data from j ∈ V_H then
    r(v) ← NumberOfReplicas(data)
    Call ForwardData(data) r(v) times
if node i ∈ V_L receives data from parent then
    Call ForwardData(data)
```

The data dissemination phase is at the heart of ProFlex, responsible for propagating properly data to the storing nodes. In the case of a uniform selection criterion, ProFlex ensures a uniform distribution of node's data among the storing nodes set. In fact, the partial view of nodes is constructed due to the distribution on each tree of a number of replicas $r(v)$ of each data. More specifically, the transmission of $r(v)$ replicas by the root i of each tree t_i will guarantee the storage of $v = \sqrt{|S^i_{aggr}|}$ data at each storing node of this tree.

Algorithm 3 shows the main actions a node must perform when it produces or receives a data packet. Initially, when a *L-sensor* node produces a data packet, it sends its packet to the *H-sensor* node that is the root of the tree the *L-sensor* node belongs to.

When a *H-sensor* node receives a data packet, three actions take place. First, it forwards the received data packet to every other *H-sensor* node that is within

its communication radius (only if the received packet was not from another *H-sensor* node). Note that, this is done using the long range communication links. By forwarding packets to other trees in the network, we intend to increase the algorithm resiliency to local failures and catastrophic events on a given region.

Second, a *H-sensor* node calls **NumberOfReplicas** (Algorithm 4) to determine how many replicas it must produce in order to forward to its children. Such computation should ensure that nodes belonging to the set of storing nodes receive with high probability $|v|$ distinct data from other nodes. As for the *partial view size*, the number of replicas is computed based on the set size of the storing nodes. As shown in [4], when $|S| = n$, then $r(v) = n \ln(\frac{n}{n-v})$; and when $|S| \neq n$, then $r(v) = n \ln n$. In particular, as described in [4], the *partial view size* $|v| = \sqrt{n}$ offers a good tradeoff between resilience and resource consumption and makes $r(v) \approx \sqrt{n}$, when $|S| = n$. Hereafter, we describe how we dynamically configure it in our proposed algorithm.

Algorithm 4: NumberOfReplicas(*data*)

Input: Data packet
Output: Number of replicas

if *data.priority = high priority* **then**
  return $\sqrt{|S^i_{aggr}|}$
if *data.priority = normal priority* **then**
  return $\frac{\sqrt{|S^i_{aggr}|}}{2}$
if *data.priority = low priority* **then**
  return $\frac{\sqrt{|S^i_{aggr}|}}{3}$

Priority policies and $r(v)$ computation. One of the foremost contributions proposed in this work is the use of *priority policies* to determine how many replicas a given data should have. The main idea behind this choice is that a higher priority packet certainly deserves a higher number of replicas when compared with a packet with a lower priority, i.e., a lower priority packet should not consume as many resources of the network as one with a higher priority. To accomplish this, we adopt a *priority policy* based on three priority levels: high, normal and low. Packets of each priority are then replicated $r(v)$ times, where $r(v)$ is defined according to (i) the formalization given by Supple (as described above); and (ii) the partial view sizes previously computed to each tree t_i: $v = \sqrt{|S^i_{aggr}|}$. Higher priorities packets are replicated $r(v) = \sqrt{|S^i_{aggr}|}$ times ($|S^i_{aggr}|$ is calculated in Algorithm 2 for each tree). Packets with normal priorities are replicated $r(v) = \frac{\sqrt{|S^i_{aggr}|}}{2}$ times, while low priority packets are replicated $r(v) = \frac{\sqrt{|S^i_{aggr}|}}{3}$. Note that, although a priority policy based on three levels was adopted, any other policy may be employed. It is only required that for a different priority value, the procedure **NumberOfReplicas** returns a different value for $r(v)$.

Finally, after determining the number of replicas, the *H-sensor* node calls **ForwardData** (Algorithm 5) $r(v)$ times. The propagation by the *H-sensor*

node is done according to the importance factor of its left and right subtree and also to its own importance factor. Moreover, when a *L-sensor* node receives a data packet from its parent, it also calls **ForwardData** to determine whether it will forward or store the packet. It must be noted that messages are forwarded asynchronously, i.e., a *H-sensor* node does not need to finish the $r(v)$ sequential data transmission of a node to start transmitting data of another one. The algorithm naturally stops when the message is received by a leaf node in the set of storing nodes. The *partial view size* of a node, i.e., the maximum number of packets a node can store, is piggybacked on every data packet. After the *H-sensor* node knows the set size of the aggregated storing nodes $|S_{aggr}|$, it calculates the partial view size $|v|$ (Algorithm 2) and embeds this value in every replicated data packet. Using this information, a *L-sensor* node knows when its buffer reached its allowed capacity.

At the end of the data dissemination, all nodes of the set of the storing nodes will have, with high probability, a partial view size v corresponding to its tree with uniformly disseminated data.

Algorithm 5: ForwardData(*data*) procedure

Input: Data packet

Pick at random uniformly $x \in [0, I_l(i) + I(i) + I_r(i)]$
if $x < I_l(i)$ **then**
  Send *data* to left child
if $I_l(i) \leq x \leq I(i)$ **then**
  Store *data* in own view
if $I(i) + I_l(i) < x$ **then**
  Send *data* to right child

3.2 Discussion

Existing protocols, in special Supple, suffer from well-known problems like hot spots closer to the sink (root), huge volume of traffic, and reliability issues. As discussed in Section 4, the use of a heterogeneous network and a protocol specially tailored to operate on it yields a series of benefits. First, its small average path length means that during the gathering phase, the information can be timely retrieved. Moreover, the overhead on the nodes closer to the root of the tree is drastically reduced. Finally, although the use of more powerful nodes (even a small number) may result in an increase on the network's cost design, heterogeneous networks provide a level of resilience that is difficult to achieve with homogeneous ones.

The use of data priority has also its benefits. Basically, by treating the data produced by the network according to its importance means that a data packet will not consume as many resources as every other data packet in the network. As shown below, reducing the number of replicas of a given packet does not lead to a great impact on ProFlex's efficiency in data gathering.

4 Performance Analysis

In order to assess the behavior of our proposed protocol, a series of simulations was performed. The simulator used was Sinalgo [7], version 0.75.3. All the results are the arithmetic mean of 33 simulations with a confidence interval of 95%. Each simulation comprises a dissemination, where each node sends its sensed data in a single data packet, and a gathering phase. In the first phase, each node executes either ProFlex or Supple for data dissemination. Then, in the gathering phase, a mobile or a static sink node will collect the data from the storing nodes. In particular, the sink will perform as many visits (in the mobile case) or send as many queries (in the static case) as necessary to get a representative amount of data of the network, i.e., getting n_l different entries of storing node's views.

4.1 Experimental Setup

In our experiment, four scenarios have been considered. In the first one, ProFlex or Supple performs the dissemination phase, and then a mobile sink visits all the nodes in the network by using a random trajectory. It is assumed a perfect transmission medium, without loss or interference. The second scenario adds to the previous one the loss of messages in the network, after the importance factor distribution phase ends. By evaluating this scenario, we show how our proposed protocol surpass Supple resilience to failures. In the third scenario, again ProFlex or Supple performs the dissemination phase, but no mobile sink performs the gathering phase. Here, after the dissemination, a static sink starts querying all data in the network. Finally, the fourth scenario also adds to the previous one the loss of messages in the network, after the end of the importance factor distribution phase 3.1.2. It is worth mentioning that in all scenarios, 33 different random choices are performed at the **ForwardData** 3.1.3 under 33 different topologies.

We evaluate the following metrics: (i) the message overhead as a function of the node's depth in the binary tree; (ii) data gathering efficiency, which is the accumulated amount of collected information after a node is visited by the mobile sink; and (iii) search query efficiency, which is the number of hops to reach a given information. In all figures, ProFlex represents our proposed protocol without the employment of data priority policies, i.e., all data packets are treated equally; ProFlex-Pr represents our proposed protocol with data priority enabled as stated in Section 3.1.3; and Supple represents the Supple protocol.

4.2 Simulation Parameters

To conceive the simulation, a fixed number of $n = 1010$ nodes was adopted, where $n_l = 1000$ *L-sensor* nodes and $n_h = 10$ *H-sensor* nodes, and only *L-sensor* nodes store data. In the case of Supple, a *H-sensor* node positioned at the center of the sensor field makes the role of a central node responsible for starting the algorithm. The nodes are randomly placed in a squared area following a uniform distribution. The average number of neighbors of a given node was fixed in 20.

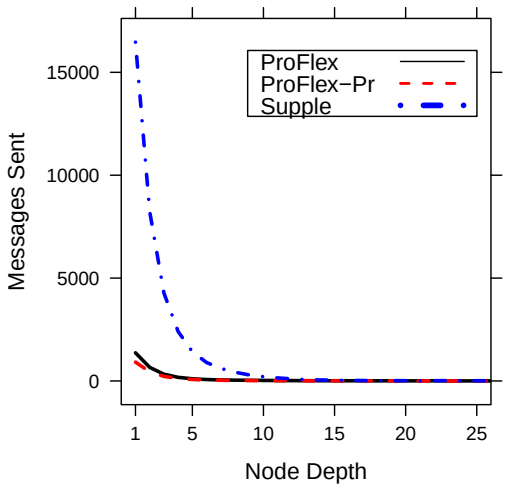

Figure 1: Average number of messages sent by a node as a function of its depth in the tree. .

Storing nodes are uniformly selected and the size of the storing nodes set is $|S| = V_L = n_l$ and $I(i) = 1$ for all nodes $i \in S$, or $I(i) = 0$, otherwise. In Supple, the view size of nodes are statically configured to $v = |\sqrt{n_l}| = |\sqrt{1000}| = 32$ entries and, consequently, $r(v) = |\sqrt{1000}| = 32$. Nevertheless, these parameters are dynamically configured in ProFlex and depend on the number of storing nodes (e.g., the number of L-sensors here) in each tree 3.1. A *size-based policy* for buffer management was employed, so the oldest entry is removed to make room for new information in the view. Moreover, when adding a new entry to the view, it is first checked if it is already present, hence no entry is added twice.

4.3 Simulated Results

4.3.1 Communication Overhead

A well-known problem in WSNs is the energy hole problem [10, 11, 18] in which nodes closer to the sink tend to consume its energy resources faster than other nodes, since they have to route packets from all other nodes in the network. In ProFlex and Supple, when a node has data to distribute to the network, it first sends the data to the root of the tree, and only then the root node will be responsible for distributing the data to the network. Hence, nodes closer to the root node tend to route more packets than other nodes as the case of the energy hole problem. Therefore, we evaluated the impact of the node's depth in the binary tree regarding the number of data messages transmitted. Fig. 4.3.1 shows the average number of data messages sent by a node as a function of its depth in the binary tree.

As expected, the closer the node is to the root, the more messages it has to send. However, in ProFlex, each *H-sensor* node is also a root node, and, consequently, more trees for data distribution are created in the network. The

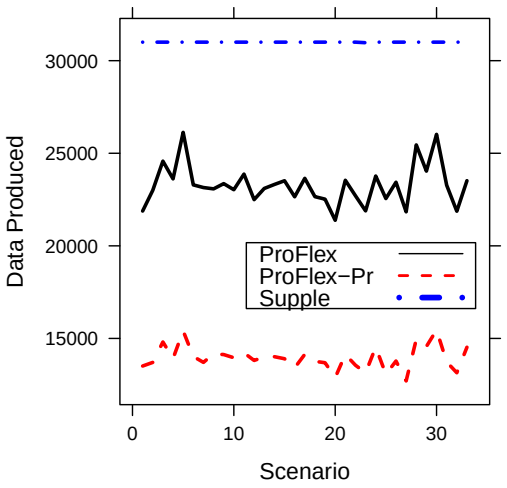

Figure 2: Number of replicated data packets in the networks as function of the scenario..

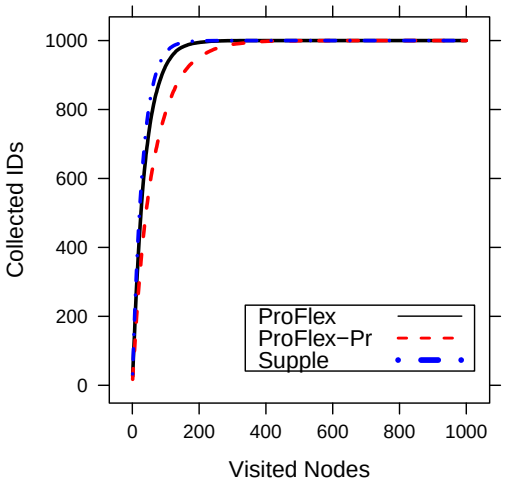

Figure 3: Amount of collected data per visited node.

overhead is, thus, distributed among the trees, alleviating the impact of the energy hole problem. It is worth mentioning that although Supple is also operating under a heterogeneous infrastructure, it is not tailored to take advantage of the features provided by this kind of network, as opposed to ProFlex. For instance, when the node's depth is 1, ProFlex and ProFlex-Pr send 91% and 94% less messages than Supple, respectively.

Using a data priority scheme also decreases the overhead over the nodes closer to the root. This is due to the amount of data packets being replicated in

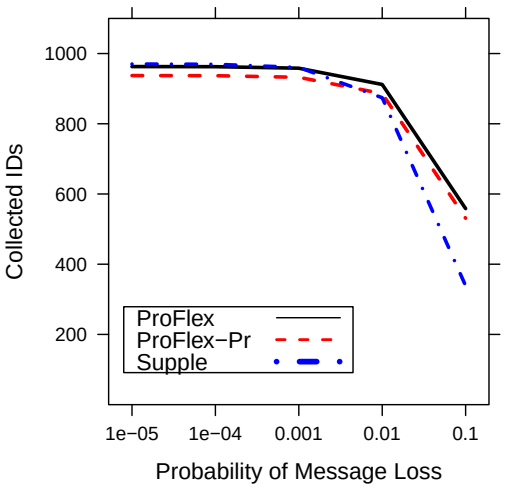

Figure 4: Amount of collected data as a function of probability of message loss.

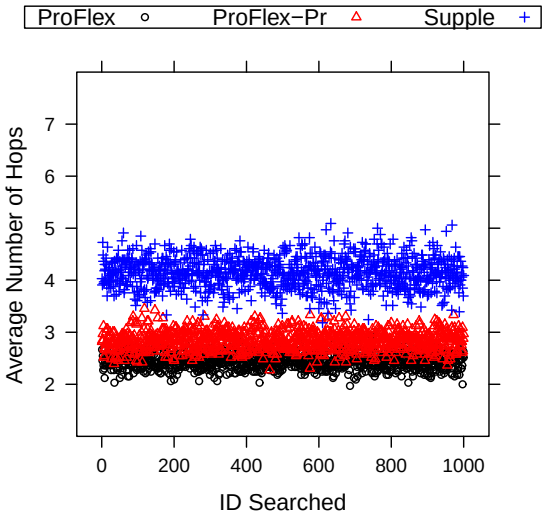

Figure 5: Average number of hops to reach a specific data.

the network (Fig. 4.3.1). In Supple, the number of replicas is a function of the size of the storing nodes set $|S|$ and nodes' view size $v = \sqrt{|S|}$, so $r(v) = \sqrt{|S|}$. Under this scheme, for a network with $S = 1000$, after the data replication phase, there are 32000 replicated data packets in the entire network (i.e., each node stores 32 data regarding other distinct nodes). Meanwhile, in ProFlex-Pr, not all packets are equally replicated. Only packets with the greatest priority will be replicated $r(v) = \sqrt{|S|}$ times. Clearly, the reduction factor depends on the *priority policy* adopted. In the scenario showed in Fig. 4.3.1, all data

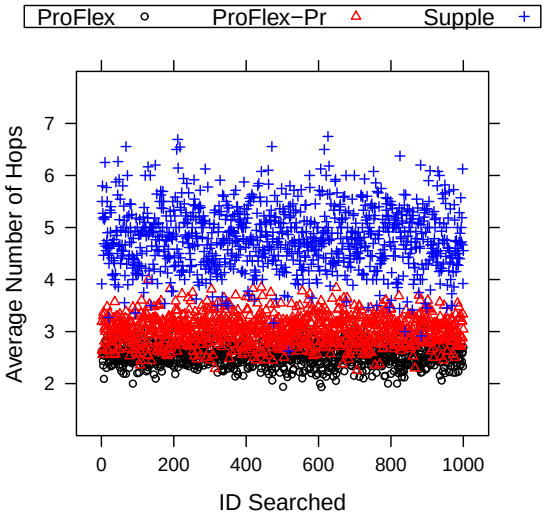

Figure 6: Average number of hops to reach a specific data in the presence of message loss equal to 10%.

packets have a priority uniformly randomly chosen between *high priority*, *normal priority*, and *lower priority*. Compared with Supple, the adopted priority policy resulted in a reduction of approximately 54% in the number of replicated data packets in the network. Note that, even when operating without data priority, ProFlex reduces in 24% the number of replicated data packets. Hereafter, we show the impact of this reduction on the data gathering and searching phase.

4.3.2 Efficiency in Data Gathering

After the dissemination of data to the network, a mobile sink placed at a random position visits the node at this position and then chooses the next position to visit, as described in [6]. When visiting a node, the sink gathers all data stored at this node. This procedure continues until all nodes have been visited. Fig. 4.3.1 shows the aggregated number of collected IDs (representing the data spread in the network) as a function of the number of visited nodes. As can be observed, Supple still presents better efficiency in data gathering, despite being closely followed by our proposed approaches. The explanation for such a behavior is that Supple still replicates a greater number of data in the network when compared to ProFlex, due to its use of a static number of replicas. For instance, Supple needs to visit about 200 nodes to get a complete view of the network, while ProFlex-Pr needs to visit about 300 nodes, a 10% increase. For a reduction in 54% in the amount of data replicated with an increase of just 10% on the number of nodes that needs to be visited, we argue that this is a good tradeoff.

To assess the resilience of our proposed protocol when compared with Supple, the former scenario was reused, but now with the introduction of message losses during the data distribution phase. Fig. 4.3.1 shows the amount of collected data after the mobile sink has visited 300 nodes as a function of the probability

of message loss. As can be observed, the use of a heterogeneous infrastructure greatly increases the resilience of our proposed approaches in the presence of message losses when compared with Supple. For instance, under a probability of message loss of 10%, Supple is able to recover about 33% of the entire data produced, whereas ProFlex is able to recover about 55% of the entire data, almost 22% more data. Thus, even requiring more visits under a reliable scenario, ProFlex and ProFlex-Pr behave much better under an unreliable scenario than Supple, making the tradeoff acceptable.

4.3.3 Efficiency in Search Query

In this scenario, there is no mobile sink for data gathering, but a static sink positioned at the center of the sensor field. After the data distribution phase, the static sink starts issuing queries for all data produced, and the number of hops to reach the data is recorded. It is worth noticing that the sink always tries to reach the data that is closer to him. Fig. 4.3.1 shows the average number of hops to reach all the data in the network. The existence of multiple trees in the ProFlex protocols, in which each tree is rooted at each *H-sensor* node, results in trees of a smaller height than a single tree as in the case of Supple. Therefore, ProFlex and ProFlex-Pr need less hops to reach a given data. Moreover, as discussed in Section 2.2, the existence of long range links reduces the average path length, and, therefore, the number of hops to reach a given node. For instance, ProFlex needs on average 2.4 hops to reach a given data, ProFlex-Pr needs 2.8 hops and Supple needs 4.1 hops. The small increase in the number of hops of ProFlex-Pr when compared with ProFlex is that the latter replicates much more packets than the former.

Here, again we assess the behavior of the proposed protocol under the presence of message losses, in the scenario with a static sink. During the data distribution phase, message losses are introduced in the network. Then, the sink starts issuing queries for all data in the network. It is also assumed that the queries issued by the sink are not lost. Fig. 4.3.1 shows the average number of hops to reach all data in the network for a probability of message loss equal to 10%. Again, ProFlex and ProFlex-Pr need less hops to reach all data in the network. For instance, ProFlex needs on average 2.5 hops to reach a given data. This represents an increase of just 4% on the average number of hops when compared with a reliable scenario. Finally, Supple needs on average 4.7 hops to reach a given data, an increase of about 12% on the average number of hops when compared with a reliable scenario.

5 Conclusion and Future Work

In this work, we presented ProFlex, a distributed data storage protocol for heterogeneous WSNs. We showed the use of a heterogeneous infrastructure greatly increases ProFlex's resilience to message losses. For instance, under the presence of a probability of message loss of 10%, ProFlex recovers about 22% more data than Supple. Additionally, ProFlex requires about 41% less hops to gather information from a static sink. Thus, ProFlex poses a significant improvement over Supple with an acceptable tradeoff. Moreover, simulation results revealed that the adoption of a new replication policy that does not treat

all packets equally, as the case of ProFlex-Pr, drastically reduces the amount of replicated packets in the network, yet without hurting efficiency. For instance, with a reduction of about 54% in the number of replicated data in the network, ProFlex-Pr needs about 10% more visits than Supple to gather all data in the network. As future work, we intend to assess the data distribution efficiency under different topologies and investigate the minimum number of powerful nodes required to achieve good results regarding collection and search efficiency.

References

[1] G. Anastasi, M. Conti, and M. Di Francesco. Data collection in sensor networks with data mules: An integrated simulation analysis. In *IEEE Symposium on Computers and Communications*, pages 1096–1102, 2008.

[2] Andre Luiz Lins Aquino and Eduardo Freire Nakamura. Data centric sensor stream reduction for real-time applications in wireless sensor networks. *Sensors*, 9(12):9666–9688, 2009.

[3] Ziv Bar-Yossef, Roy Friedman, and Gabriel Kliot. Rawms - random walk based lightweight membership service for wireless ad hoc networks. *ACM Trans. Comput. Syst.*, 26:5:1–5:66, 2008.

[4] Aline C. Viana, Thomas Herault, Thomas Largillier, Sylvain Peyronnet, and Fatiha Zaïdi. Supple: a flexible probabilistic data dissemination protocol for wireless sensor networks. In *Proc. of the 13th ACM Inter. Conf. on Modeling, analysis, and simulation of wireless and mobile systems*, pages 385–392, 2010.

[5] J. Eriksson, M. Faloutsos, and S. Krishnamurthy. Scalable ad hoc routing: the case for dynamic addressing. In *23rd Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 1108–1119, 2004.

[6] Roy Friedman, Gabriel Kliot, and Chen Avin. Probabilistic quorum systems in wireless ad hoc networks. In *In Proc. of the 38th IEEE Inter. Conference on Dependable Systems and Networks*, pages 277–286, 2008.

[7] ETHZ Distributed Computing Group. Sinalgo - simulator for network algorithms. http://dcg.ethz.ch/projects/sinalgo/, 2008.

[8] Daniel Ludovico Guidoni, Raquel Aparecida Freitas Mini, and Antonio Alfredo Ferreira Loureiro. On the design of resilient heterogeneous wireless sensor networks based on small world concepts. *Elsevier Computer Networks*, 54:1266–1281, 2010.

[9] Elyes Ben Hamida and Guillaume Chelius. A line-based data dissemination protocol for wireless sensor networks with mobile sink. In *Proc. of IEEE Int. Conf. on Communications*, pages 2201–2205, 2008.

[10] Jian Li and Prasant Mohapatra. Analytical modeling and mitigation techniques for the energy hole problem in sensor networks. *Pervasive Mob. Comput.*, 3:233–254, 2007.

[11] An-Feng Liu, Xian-You Wu, Zhi-Gang Chen, and Wei-Hua Gui. Research on the energy hole problem based on unequal cluster-radius for wireless sensor networks. *Elsevier Computer Communications*, 33(3):302–321, 2010.

[12] F. Marcelloni and M. Vecchio. A simple algorithm for data compression in wireless sensor networks. *IEEE Comm. Letters*, 12(6):411–413, 2008.

[13] Bo Sheng, Qun Li, and Weizhen Mao. Data storage placement in sensor networks. In *Proc. of the 7th ACM Inter. symposium on Mobile ad hoc networking and computing*, pages 344–355, 2006.

[14] Bo Sheng, Chiu C. Tan, Qun Li, and Weizhen Mao. An approximation algorithm for data storage placement in sensor networks. In *Proc. of the Inter. Conf. on Wireless Algorithms Systems and Applications*, pages 71–78, 2007.

[15] Yu-Chee Tseng, Wan-Ting Lai, Chi-Fu Huang, and Fang-Jing Wu. Using mobile mules for collecting data from an isolated wireless sensor network. In *39th Inter. Conf. on Parallel Processing (ICPP)*, pages 673–679, 2010.

[16] M. Vecchio, Aline C. Viana, A. Ziviani, and R. Friedman. Deep: Density-based proactive data dissemination protocol for wireless sensor networks with uncontrolled sink mobility. *Elsevier Computer Communicationl*, 33(8), 2010.

[17] Zoltan Vincze, Dorottya Vass, Rolland Vida, Attila Vidacs, and Adras Telcs. Adaptive sink mobility in event-driven multi-hop wireless sensor networks. In *Proc. of 1st Int. Conf. on Integrated Internet Ad Hoc and Sensor Networks*, pages 30–31, 2006.

[18] Xiaobing Wu, Guihai Chen, and S.K. Das. Avoiding energy holes in wireless sensor networks with nonuniform node distribution. *IEEE Trans. on Parallel and Distributed Systems*, 19(5):710–720, 2008.

Contents

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 Orsay Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399